Videojuegos

Curso de Diseño y Programación

Nº 9 5,99 euros



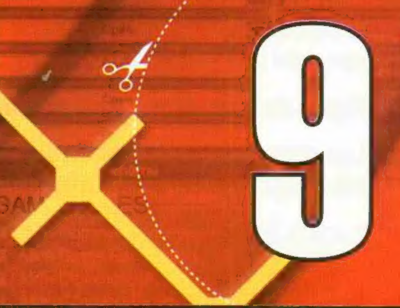
Funciones especiales para ***controlar colisiones***

Mover los modelos con ***Character FX***

Creación de música con ***Cool Edit Pro***

MULTIPLAYER SETUP

PLAYER SETUP





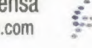




Curso práctico de Videojuegos

AUTOR DE LA OBRA
Marcos Medina
DIRECCIÓN EDITORIAL
Eduardo Toribio
etoribio@iberprensa.com
COORDINACIÓN EDITORIAL
Eva-Margarita García
eva@iberprensa.com
DISEÑO Y MAQUETACIÓN
Antonio Gª Tomé
PRODUCCIÓN
Marisa Cogorro
SUSCRIPCIONES
Tel: 91 628 02 03
Fax: 91 628 09 35
suscripciones@iberprensa.com
FILMACIÓN: Fotpreim Duvial
IMPRESIÓN: Gráficas Don Bosco
DUPLICACIÓN CD.ROM: M.P.O.
DISTRIBUCIÓN
S.G.E.L.
Avda. Valdelaparra 29 (Pol. Ind.)
28108 Alcobendas (Madrid)
Tel.: 91 657 69 00

EDITA: Iberprensa
www.iberprensa.com



CONSEJERO
Carlos Peropadre

REDACCIÓN, PUBLICIDAD Y ADMINISTRACIÓN
C/ del Río Ter, 7 (Pol. Ind. "El Nogal")
28110 Algete (Madrid)
Tel.: 91 628 02 03
Fax: 91 628 09 35
(Añada 34 si llama desde fuera de España.)

DEPÓSITO LEGAL: M-35934-2002
ISBN: Coleccionable: 84 932417 2 5
Tomo 1: 84 932417 3 3
Obra Completa: 84 932417 5 X

NOTA IMPORTANTE:
Algunos programas incluidos en los CD de "Programación y Diseño de Videojuegos" son versiones completas, pero en otros casos se trata de versiones demo o trial, versiones de evaluación que Iberprensa quiere ofrecer a nuestros lectores. No se trata en ningún caso de las versiones comerciales de los programas, y las hemos incluido para dar al lector la oportunidad de conocer y probar esos programas y que así pueda decidir posteriormente si desea o no adquirir las versiones comerciales de cada uno.

Aprende divirtiéndote

Bienvenidos a **Programación y Diseño de Videojuegos**, la primera obra coleccionable cuyo objetivo es formar al alumno en las principales técnicas relacionadas en el desarrollo completo de un videojuego.

A lo largo de la obra el lector aprenderá programación a nivel general y a nivel específico con ciertas herramientas y lenguajes, aprenderá a trabajar con aplicaciones de retoque de imagen y también de diseño 3D y animación. Descubrirá las aplicaciones profesionales más importantes de audio y conocerá la historia de lo que se denomina "la industria del videojuego", los últimos 20 años, los juegos que marcaron un avance, sus creadores y en general la evolución del videojuego.

Pero además, esta obra tiene un segundo objetivo, desarrollar y potenciar la creatividad del lector, nosotros a lo largo de las diferentes entregas pondremos las bases y tú pondrás tu ingenio, tu creatividad y tu capacidad de mejorar.

Comienza aquí un viaje de 20 semanas articulado en 400 páginas y 20 CD-ROMs cuya finalidad es proporcionar las bases mínimas para después cada uno continuar su camino.

Recuerda que para alcanzar el éxito necesistas cumplir tres condiciones: que te gusten los juegos, poseer cierta dosis de creatividad y finalmente capacidad de estudio.

Una la cumples seguro.

sumario



PARA ENCUADERNAR LA OBRA:

- *Para encuadernar los dos volúmenes que componen la obra "Programación y Diseño de Videojuegos" se pondrán a la venta las tapas 1 y 2.*
- ***Tapas del volumen 1 ya a la venta.***
- ***Los suscriptores recibirán las tapas en su domicilio sin cargo alguno como obsequio de Iberprensa.***

SERVICIO TÉCNICO:
Para consultas, dudas técnicas y reclamaciones Iberprensa ofrece la siguiente dirección de correo electrónico: games@iberprensa.com

PETICIÓN DE NÚMEROS ATRASADOS:
El envío de números sueltos o atrasados se realizará contra reembolso del precio de venta al público más el coste de los gastos de envío. Pueden ser solicitados en el teléfono de atención al cliente 91 628 02 03

Visualizando el terreno de juego

Vamos a codificar nuestro juego estudiando los diferentes procedimientos que intervienen en él de forma independiente por medio de la implantación de funciones.

Debido a la extensión del código del juego, sólo explicaremos las partes más importantes, suficiente para comprender todo el funcionamiento. Aun así, en el código fuente se puede encontrar cada una de las partes comentadas con detalle, para que no haya ninguna pérdida en su seguimiento.

Así que empezaremos por un módulo realmente vistoso y espectacular: la creación del entorno de juego y la implementación de todos los aspectos de la cámara.

Las funciones especiales de vídeo y audio las implementaremos en el módulo "funcpantaudio.bb" del juego. Crearemos entonces una función llamada *Crear_entorno()*, que nos servirá para generar el terreno de juego a partir de un mapa de alturas y de sus texturas elegidas en el módulo "menu.bb" (el cual veremos con detalle en otra ocasión), así como de la creación de las luces ambientales, el agua y el cielo.

A medida que definamos variables las iremos incluyendo en el módulo de definiciones "definiciones.bb".

Un buen método de trabajo sería crear las plantillas de todos los módulos y tenerlas cargadas en el editor del Blitz3D. De esta forma, se puede pasar de un módulo a otro con facilidad para añadir datos.

Código 1. Variable global "Tipo_juego"

```
If Tipo_juego=1
 Repeat
  Cls
  Fondo_menu()
  DrawImage logo,20,GraphicsHeight()-100
  DrawImage titulo(5),(GraphicsWidth()/2)-(ImageWidth(titulo(5))/2),20
  Flip
  Locate (GraphicsWidth()/2),150
  Tipo_terreno=Input$(">")
  If Tipo_terreno=0 Then Return
 Until Tipo_terreno>0 And Tipo_terreno<3

 Select Tipo_terreno
  Case 1:
    Terreno$="c:\zone of fighters\gfx\terreno1.bmp"
    textura0=LoadTexture("c:\zone of fighters\gfx\textura1.bmp")
    textura1=LoadTexture("c:\zone of fighters\gfx\textura2.bmp")
  Case 2:
    terreno$="c:\zone of fighters\gfx\terreno2.bmp"
    textura0=LoadTexture("c:\zone of fighters\gfx\textura3.bmp")
    textura1=LoadTexture("c:\zone of fighters\gfx\textura4.bmp")
 End Select
EndIf
```

(■) ELIGIENDO EL TERRENO

Antes de crear el entorno, debemos codificar la parte donde el jugador elige el tipo de terreno. Está elección la implementamos en la función "panel de opciones" del menú de juego.

No entraremos en detalles sobre la estructura del menú porque la veremos en una próxima entrega. Lo fundamental que debemos saber aquí es cómo el ordenador lee un mapa u otro con sus texturas dependiendo de la opción elegida. El jugador dispone de dos tipos de zonas de combates con dos terrenos (mapas de alturas) diferentes por cada zona. Así que lo primero es preguntar qué tipo de juego desea.

(■) ZONA GENERADA ALEATORIAMENTE

En este tipo de juego, el ordenador coloca aleatoriamente, siguiendo unos patrones definidos, todo el decorado, plantas y animales sobre dos terrenos diferentes a elegir. El tipo de juego lo almacenamos en la variable global "Tipo_juego" (Ver tabla adjunta, Código 1).

Básicamente, lo que hacemos es pedir al usuario qué terreno y texturas debemos cargar para el tipo de juego 1 (zona generada aleatoriamente) con la instrucción "Input". Pero antes mantenemos el fondo dibujado llamando a la función *"Fondo_menu()"* e imprimimos la imagen con la frase "Elegir terreno" contenida en título (5) de la matriz de títulos.

Una vez obtenido el dato pasamos a una estructura *"Select .. Case"* en donde, dependiendo del tipo de terreno, asignamos a la variable global *"terreno$"* la situación del fichero del mapa de alturas "terreno1.bmp" o "terreno2.bmp", así como las texturas en "textura0" y "textura1".

(■) ZONA EDITADA

Si el jugador elige la opción de cargar una zona editada previamente con el editor de zonas, debemos obtener el registro que contiene el tipo de terreno que lleva dicha zona. Pero antes, es necesario realizar una serie de operaciones para obtener este dato. Aprovechamos entonces para explicar el proceso necesario para obtener el fichero de datos que contiene toda la zona de combate elegida. En primer lugar disponemos de una matriz llamada *"zona_elegida$"*, la cual contiene todas las capturas de las zonas editadas por el editor. Pero para poder dimensionar con un valor fijo debemos contar cada una de las capturas del disco. Vamos e entonces a crear dicha matriz. Esta operación la realizamos en el módulo "definiciones.bb":

```
dir=ReadDir("c:\zone of
fighters\gfx\datos\")
c=0
Repeat
 f$=NextFile(dir)
 If Left(f$,4)="Img_"  c=c+1
Until f$=""
Dim zona_elegida$(c)
```

Leemos el directorio que contiene las capturas, a continuación recorremos cada fichero e incrementamos el contador "c" cada vez encuentre un archivo que empiece por "Img_". Para terminar, dimensionamos la matriz con el valor del contador.

De vuelta al módulo "menu.bb", haremos la misma operación pero al contrario, es decir, leemos cada fichero con la captura pero para introducir su nombre en la matriz *"zona_elegida$"*:

```
If Tipo_juego=2
  dir=ReadDir("c:\zone of
  fighters\gfx\datos\")
  c=0
  Repeat
   f$=NextFile(dir)
   If Left(f$,4)="Img_"
      c=c+1
      zona_elegida$(c)=f$
   EndIf
  Until f$=""
```

Seguidamente, visualizamos cada captura, previamente reducida a la mitad, en la pantalla. Para ello, entramos en un bucle en donde el jugador, mediante los cursores, pasa de un fichero a otro.

Durante la elección, el índice de la matriz del fichero correspondiente lo almacenamos en la variable local *"cont"*, para luego saber qué fichero cargar. Salimos del bucle cuando se haya pulsado "Return"con nuestro valioso dato almacenado en *"cont"* (Ver tabla Código 2).

Una vez obtenido el nombre del fichero con la zona elegida, debemos cambiar el nombre del fichero contenido en la matriz *"zona_elegida$"*, ya que éste sólo muestra una captura y no los datos que necesitamos. Por lo tanto, aprovechando que diseñamos un nombre de fichero similar para las capturas y los datos, sólo tendremos que cambiar el encabezamiento "Img_" por "Dat_" y la extensión ".bmp" por ".zof".

```
zona_de_combate$=""
zona_de_combate$=zona_elegida$(cont)
```

```
zona_de_combate$=Replace
(zona_de_combate$,".bmp",".zof")
zona_de_combate$=Replace
(zona_de_combate$,"Img_","Dat_")
```

A continuación, leemos el fichero con los datos y el primer bloque de registros.

Rechazamos todos los registros (referentes a objetos) menos el último, que contiene el tipo de terreno utilizado para esa zona. Almacenamos este dato en la variable global *"Tipo_terreno"*:

```
fichero=ReadFile("c:\zone of
fighters\gfx\datos\"+zona_de_
combate$)
Null1$=ReadString(fichero)
null2=ReadByte(fichero)
null3=ReadFloat(fichero)
null4=ReadFloat(fichero)
null5=ReadFloat(fichero)
null6=ReadFloat(fichero)
null7=ReadFloat(fichero)
Tipo_terreno=ReadByte(Fichero)
```

Una vez que hayamos obtenido el tipo de terreno utilizamos la misma operación que el tipo de

Código 2. Variable local "cont"

```
cont=1
ze=LoadImage("c:\zone of fighters\gfx\datos\"+zona_elegida$(cont))
ResizeImage ze,GraphicsWidth()/2,GraphicsHeight()/2
Repeat
 Cls
 Fondo_menu()
 If KeyDown(205)
    cont=cont+1
    If cont=c+1 Then cont=1
    ze=LoadImage("c:\zone of
fighters\gfx\datos\"+zona_elegida$(cont))
    ResizeImage ze,GraphicsWidth()/2,GraphicsHeight()/2
 EndIf
 If KeyDown(203)
    cont=cont-1
    If cont=0 Then cont=c
    ze=LoadImage("c:\zone of fighters\gfx\datos\"+zona_elegida$(cont))
    ResizeImage ze,GraphicsWidth()/2,GraphicsHeight()/2
 EndIf
 DrawImage ze,(GraphicsWidth()/2)-(ImageWidth(ze)/2),
 (GraphicsHeight()/2)-150
 DrawImage titulo(7),(GraphicsWidth()/2)-(ImageWidth(titulo(7))/2),20
 Flip
Until KeyDown(28)
FreeImage ze
```

juego anterior para cargar el mapa de alturas y las texturas.

(■) CREANDO EL ENTORNO

Ya tenemos el mapa de alturas y las texturas correspondientes. Así que ya podemos crear el entorno del juego. Volvamos a la función *"Crear_entorno()"* del módulo "funcpantaudio.bb".

(■) CREANDO LAS LUCES

Para poder ver nuestro mundo 3D es necesario iluminarlo. Utilizaremos el comando "AmbientLight" para crear una tonalidad de luz general, la cual afectará por igual a los puntos de cada objeto. Siempre es necesario aplicarla antes de utilizar otras luces. Después crearemos una luz artificial que usaremos como si fuera el Sol.

```
Function Crear_entorno()
variacion_luz#=.05
   luz#=200
   AmbientLight luz#,luz,luz
    light=CreateLight()
   RotateEntity light,45,45,0
   LightColor light,150,150,150
        .....
```

(■) CREANDO EL TERRENO DE JUEGO

Una vez iluminado, podemos crear el terreno a partir del mapa de alturas leyendo el fichero contenido en *"terreno$"*. Según el tipo de terreno, lo escalamos adecuadamente, activamos o no el sombreado según la variable global "shading" y procedemos a darle un número de polígonos según "Detalle_terreno" para crear la forma, además de activar el sistema dinámico de detalle (LOD). Posteriormente, suavizamos los bordes o antialisado según "antialisado ("True/False")".

```
terreno1=LoadTerrain(terreno$)
Select Tipo_terreno
  Case 1: ScaleEntity terreno1,
  30,350,30    ; X=30 x 512
  =15360  Y=350  Z=15360
  Case 2: ScaleEntity terreno1,
   30,380,30
  End Select
  TerrainShading terreno1,Shading
  TerrainDetail terreno1,
  Detalle_terreno,True
  AntiAlias antialisado
```

El siguiente paso será texturizar el terreno con las texturas contenidas en las variables globales "textura0" y "textura1". La primera textura servirá de piel y dará al terreno los diferentes matices de color según la zona. Como sabemos, la forma que tiene Blitz3D de texturizar un terreno es cubrirlo completamente mediante la repetición continua de la textura (tileado). En esta repetición se utiliza un tamaño de textura por defecto de 1, es decir, el tamaño de una cuadrícula de la malla. Si queremos entonces cubrir por completo el terreno sin la utilización del tileado, debemos escalarla a su tamaño original, es decir, en nuestro caso 512 x 512.

```
PositionTexture textura0,0.,0
ScaleTexture textura0,512,512
EntityTexture terreno1,
textura0,0,0
```

La segunda se utilizará para añadir más realismo a la superficie creando imperfecciones como grietas o rugosidades. Ambas texturas aparecerán mezcladas, predominando la situada debajo. Para esta textura sí podemos jugar con el porcentaje de tileado, variando el escalado. Unas dimensiones de 64 darán buen resultado, pero a medida que la cámara se eleva se notarán más las repeticiones. Así que utilizaremos un valor intermedio de 128 x 128; es decir, se repetirá 16 veces (4 x 4). Luego, la rotamos 45 grados para disimular aún más la repetición.

```
PositionTexture textura1,0.,0
ScaleTexture textura1,128,128
RotateTexture textura1,45
EntityTexture terreno1,
textura1,0,1
```

(■) CREANDO LA URNA, EL CIELO ARTIFICIAL Y EL AGUA

En cuanto al diseño, dijimos que el terrario donde se desarrolla el juego está cubierto por una gran urna. Ésta no es más que un cubo situado de tal forma que engloba todo el terreno.

Conociendo el tamaño del terreno y su posición relativa a la posición 0,0,0, podemos situar perfectamente la urna:

```
urna=CreateCube()
EntityTexture urna,textura_urna
FlipMesh urna
ScaleMesh urna,7500,3000,7500
PositionEntity urna,
7500,1000,7500
```

La sentencia "FlipMesh urna" se utiliza para dar la vuelta a todas las caras del cubo que forman la urna para poder mostrar la textura hacia el interior. Si no lo hiciésemos, no veríamos textura alguna.

(■) CREANDO EL CIELO Y EL AGUA

El cielo es realmente un plano situado 1500 unidades por encima del terreno. Una vez creado lo ocultamos o mostramos dependiendo de si lo tenemos activado desde menú con la variable global "Cielo":

```
mesh_cielo=CreatePlane()
ScaleEntity mesh_cielo,80,20,80
PositionEntity mesh_cielo,0,1500,0
RotateEntity mesh_cielo,180,0,0
ScaleTexture textura2,512,512
EntityTexture mesh_cielo,textura2
If Cielo=True
   ShowEntity mesh_cielo
 Else
   HideEntity mesh_cielo
EndIf
```

En el módulo "Fjuego.bb" tenemos una función para actualizar todo el entorno durante la partida llamada *"Actualiza_Entorno()"*.

En esta función realizamos operaciones como mover el cielo o generar el paso del día a la noche y viceversa. Para mover el cielo realizaremos

una rotación continua de la textura:

```
If Cielo=True
  If angulo_textura#>
  360 Then angulo_textura#=0
     angulo_textura#=
     angulo_textura#+.05
     RotateTexture textura2,
     angulo_textura#
  EndIf
EndIf
```

Y para el paso del día a la noche, rotamos el Sol a la vez que disminuimos la luminosidad tanto de él como del ambiente entre los valores 10 y 200:

```
If Dia_Noche=True
  rotate_luz#=rotate_luz+.05
  RotateEntity light,rotate_luz#,
  rotate_luz,0
  If luz#<10 variacion_luz=
  (variacion_luz)*-1
  If luz#>200 variacion_luz=
  (variacion_luz)*-1
  luz#=luz-variacion_luz#
  AmbientLight luz#,luz,luz
EndIf
```

Volvamos a la función *"Crear_entorno()"* del módulo "funcpantuadio.bb". Para crear el agua debemos saber primero si se ha activado o no la opción del reflejo, ya que dependiendo de una u otra opción, se creará a partir de un plano infinito o de un cubo escalado.

Para poder implementar el reflejo es necesario crear un plano, sin embargo este plano abarcará todo el terreno. La creación de un mirror provoca la bajada del rendimiento del juego en un 50 % ya que prácticamente se renderiza el doble de polígonos.

```
If Reflejo=True
   agua=CreatePlane()
   mirror=CreateMirror()
  Else
   agua=CreateCube()
   Select Tipo_Terreno
    Case 1: ScaleMesh agua,
    4400,4,9000
    Case 2: ScaleMesh agua,
    9000,4,9000
   End Select
 EndIf
```

Una vez creada el agua la posicionamos, le asignamos un color y modificamos el canal alfa para darle un poco de transparencia.

```
PositionEntity agua,8000,25,7000
EntityColor agua,92,64,107
EntityColor agua,40,73,162
EntityAlpha agua,.6
```

(■) CREANDO LA CÁMARA

Para poder observar el entorno es necesario la creación de una cámara, la cual podremos situar en diferentes lugares dependiendo del tipo de vista que queramos: primera persona, tercera persona, cenital o de seguimiento.

Primero, será la variable global *"camara"* la que utilizaremos para manipular la cámara, a continuación es necesario asignarle un campo de visión para delimitar hasta dónde podemos ver. Este concepto es importante porque todo lo que queda detrás del campo de visión no es renderizado. Modificando el "zoom", obtendremos algunos resultados interesantes. Utilizando valores de alejamiento es posible obtener un resultado muy parecido a una lente de ojo de pez. Los valores para asignar un zoom van desde 0 hasta 1, así que podríamos utilizar, para lograr este efecto, 0.54 en la variable global "zoom".

```
cam= CreateCamera()
camara=cam
CameraRange camara,0.1,Rango_vision
CameraZoom camara,zoom
```

Otro aspecto configurable que podemos aplicar a la cámara es la niebla o "fog". Dependiendo del terreno, aplicaremos un color para la niebla u otro, así como para el fondo de la cámara.

Además debemos definir el rango de visualización de la niebla, es decir, lo lejos o cerca del observador. Como ya comentamos en su día, definir el rango de la niebla por delante del rango de visualización de la cámara disimula el corte que sufre el campo de visión (Ver tabla Código 3).

Hemos completado un paso muy importante en la creación de nuestro juego. El siguiente punto a tratar será el estudio del módulo principal y de cómo mostrar en pantalla los indicadores de juego así como el radar de combate.

Código 3. Definición del rango de niebla

```
If Niebla=True
   Select Tipo_terreno
    Case 1:
     CameraFogMode camara,niebla
       CameraFogRange camara,0.0,Rango_vision-100
     CameraFogColor camara,53,87,78
     CameraClsColor camara,53,87,78
    Case 2:
       CameraFogMode camara,niebla
       CameraFogRange camara,0.0,Rango_vision-100
     CameraFogColor camara,130,171,230
     CameraClsColor camara,130,171,230
    End Select
  Else
   CameraRange camara,0.1,20000
EndIf
PositionEntity camara,7624,93,15026
```

En el próximo número...

... seguiremos analizando el código de "Zone of Fighters" empezando por el gestor *de principal*.

Grafismo 3D. Animando nuestro protagonista con **Character FX**

Vamos a estudiar cómo conseguir dar movimientos a nuestros modelos 3D. Para ello utilizaremos la aplicación Character FX, que nos permitirá conseguir movimientos con gran eficiencia.

Aunque, en principio, en el juego "Zone of Fighters" no implementaremos la bionave con movimiento alguno, su aplicación nos ayudará a comprender el funcionamiento de esta herramienta para poder animar las plantas y animales de la zona de combate en próximas entregas.

CONOCIENDO LA INTERFAZ DE USUARIO

Antes de empezar, vamos a describir las principales partes de la interfaz de usuario de Character FX para tener una idea de su filosofía de funcionamiento.

Para empezar, Character FX trabaja en dos modos diferentes: modo de creación de esqueletos *(objects)* y modo animación *(animation)*. Para seleccionar un modo u otro debemos elegirlo con la paleta de selección de herramientas situada en la ventana principal de la aplicación. En la figura 1 se muestran las partes de esta ventana.

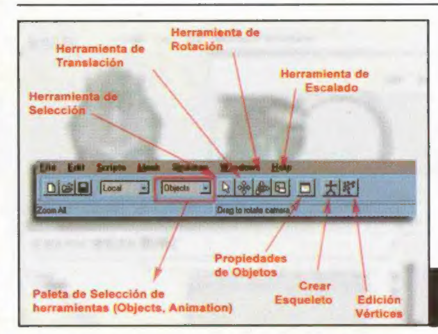
Todo el trabajo que realicemos se muestra en los "Viewports" o vistas. Al ejecutar el programa se activará una vista por defecto, "perspective: 1". Podemos abrir todas las vistas que queramos con la opción "New Viewport" en el menú "Windows" de la ventana principal. Cada una de estas vistas tiene su propio menú de opciones, que veremos a medida que trabajemos con la animación de nuestro modelo. Volviendo a la ventana principal, nos encontramos con una serie de iconos que representan las herramientas disponibles para cada modo de trabajo.

HERRAMIENTAS EN EL MODO "OBJECTS"

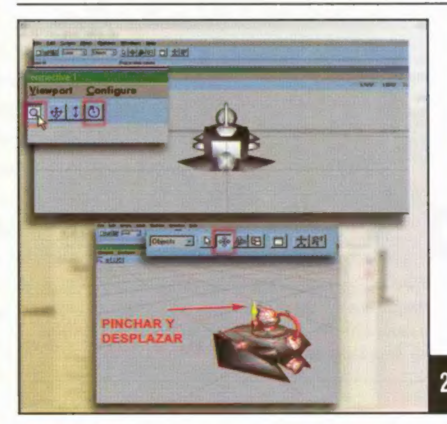
Para trabajar con los modelos y crearles un esqueleto, disponemos de 7 herramientas: herramienta de selección, herramienta para mover, herramienta para rotar, para cambiar de tamaño, herramienta para editar las propiedades de los modelos, herramienta para crear el esqueleto mediante uniones de huesos y herramienta de asignación de vértices para la posterior animación.

Herramienta de selección

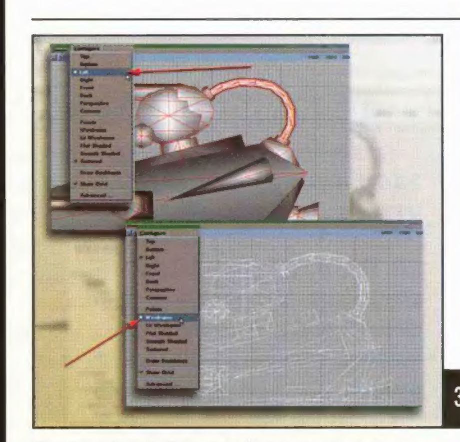
Esta opción nos permitirá seleccionar vértices, uniones u objetos completos. Si pulsamos el botón derecho del ratón sobre el icono aparecerá una pequeña ventana de opciones, en la cual podemos elegir qué clase de selección realizaremos. Para seleccionar un objeto debemos hacer clic sobre él en cualquier vista que tengamos abierta. Si queremos sumar más de una selección, seleccionamos mientras mantenemos pulsada la tecla *CTRL* y para eliminar selección la tecla *ALT*. Para borrar cualquier selección haremos *CTRL + Supr*.



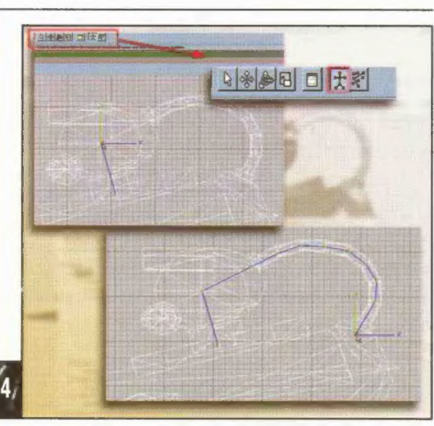
Esquema de las partes de la ventana principal de herramientas.



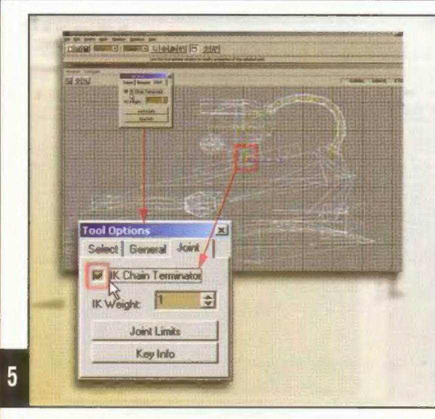
Procedimientos para preparar el modelo para la construcción del esqueleto.



En el modo wireframe podemos ver mejor nuestro trabajo de construcción del esqueleto.



Con sólo hacer clic, creamos un nuevo punto de unión.



Para realizar diferentes esqueletos independientes dentro de un mismo modelo, es necesario asignar, a cada uno de ellos, una unión que sirva de final en la cadena jerárquica.

DEFINICIÓN

▸ CINEMÁTICA INVERSA

Mediante la cinemática inversa se consiguen movimientos realistas de un modelo complejo. Consiste en aplicar más prioridad a la unión del esqueleto situada en la posición más baja de su jerarquía, de tal forma que, al mover este punto, se mueven todos los demás situados a un nivel mayor.

■ Herramientas de movimiento, rotación y escalado

Para mover una unión o un objeto debemos seleccionarlo primero. Una vez seleccionado se visualizará en su punto de pivotaje los ejes XYZ de coordenadas. Si queremos mover el objeto hacia la derecha o izquierda movemos el ratón haciendo clic sobre la flecha de color roja (Eje X). En la flecha verde (Eje Y) para subir o bajar y en la flecha azul (Eje Z) para acercar o alejar. Para mover libremente, pulsamos sobre la esfera situada en la base de los ejes. Para rotar o escalar el objeto se utilizan los mismos procedimientos. En estas herramientas podemos activar las opciones pulsando con el botón derecho sobre su icono. Observamos que podemos mover, rotar y cambiar el tamaño proporcionando valores exactos.

■ Editando las propiedades de un objeto

Una vez seleccionado un objeto o una unión, podemos editar sus propiedades con esta opción. Para un objeto, podemos asignarle un nombre y un color. Si elegimos una unión las opciones son más amplias: desde asignar el peso que tendrá dicha unión dentro del esqueleto hasta definirle unos límites de movimientos. Explicaremos más a fondo estas cualidades cuando creemos el esqueleto para la bionave.

■ Herramienta para crear el esqueleto mediante uniones

Esta herramienta nos permitirá crear un esqueleto completo a partir de la suma de puntos de unión. Con un simple clic en la vista pondremos un punto; si creamos otro en otro sitio, se unirá al anterior y así sucesivamente hasta formar el esqueleto.

Todas las herramientas de selección, movimiento y rotación aplicadas a las uniones se comportan de la misma manera que con los objetos.

■ Herramienta para asignar y editar vértices

Al activar esta herramienta, el modelo 3D se cubre de vértices en forma de cuadraditos de color verde. Estos vértices se utilizan para asignar una influencia a cada punto de unión; es decir, la unión seleccionada al moverse, afectará a la parte del objeto cuyos vértices hayan sido asignados a dicho punto. Además, a cada asignación le podemos aplicar un peso, el cual definirá la cantidad de influencia que ese punto tendrá en los vértices. Un valor (peso) de 0 indicará una influencia nula.

HERRAMIENTAS EN EL MODO "ANIMATION"

Una vez creado el esqueleto, pasamos al modo de animación para aplicarle movilidad. Encontramos herramientas para mover y rotar los puntos de unión, los cuales afectarán a todo el esqueleto según las asignaciones dadas en el modo "Objects". Además disponemos de una herramienta para mover el esqueleto con el método de cinemática inversa que nos permitirá realizar animaciones más realistas.

EJEMPLO PRÁCTICO. ANIMANDO LA BIONAVE DE COMBATE

Para empezar debemos importar el modelo de nuestra bionave de combate. Vamos a utilizar la que modelamos en su día con Milkshape3D. Si no se tiene a mano, en la carpeta "Extras" del CD que se adjunta se encuentra una copia del modelo 3D en formato .OBJ ("bionave.obj") y su textura ("textura_bionave.png").

Para importar el fichero "bionave.obj" elegimos la opción "File / Import". Una vez cargada la bionave, vemos cómo es mostrada de espaldas en la vista principal. Para ajustar de una forma rápida el modelo a la ventana pulsamos sobre el icono de la lupa una vez. A continuación, movemos la vista pulsando el icono de rotar la cámara y moviendo el ratón haciendo clic sobre cualquier lugar de la cuadrícula de la vista (Fig 2).

También podemos mover la cámara en los ejes X y Z (hacia la derecha e izquierda y hacia delante y atrás) mediante el icono y en el eje Y (arriba y abajo) activando el icono. Una vez situada la vista como se muestra en la figura 2, seleccionamos la herramienta de traslación de la ventana principal y pulsamos sobre el modelo. Al aparecer los ejes, movemos la bionave hacia arriba pulsando sobre la flecha amarilla (eje Y) hasta que quede por encima de la cuadrícula (Fig. 2).

CREANDO EL ESQUELETO

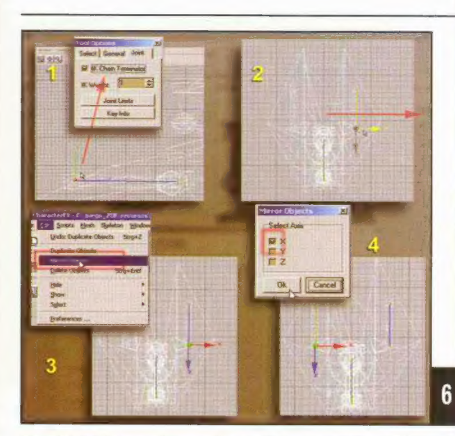
Para crear el esqueleto vamos a situar la vista en proyección lateral izquierda. Elegimos la opción "Left" en el menú "Configure" de la ventana de vista. Acercamos la bionave hacia la cámara mediante el ratón con la herramienta de "Zoom" activada.

Deseleccionamos el modelo activando la herramienta de selección y haciendo clic en cualquier lugar de la vista. Después, conmutamos el modo de representación a "Wireframe" en "Configure" para poder ver mejor las uniones que dibujemos (Fig. 3)

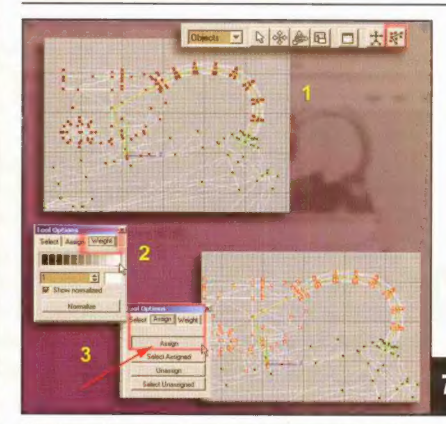
Nuestro propósito es hacer girar la cabeza del protagonista. No debemos olvidar que el casco tiene un tubo flexible que está unido al fuselaje por uno de sus extremos. Así que al mover la cabeza hacia los lados el tubo lo hará también, mientras sigue sujeto al cuerpo de la nave. Necesitaremos entonces crear un esqueleto que recorra el tubo, llegue hacia el centro de la cabeza y siga hasta la base del cuello. Activamos la herramienta de creación de uniones y hacemos clic en el centro del cuello y otro clic en el centro de la cabeza. Observamos cómo los dos puntos se unen. Ya hemos creado un pequeño esqueleto. Vamos a seguir uniendo puntos para recorrer el tubo. Colocaremos un punto nuevo en cada unión de los polígonos del tubo como se muestra en la figura 4.

Además de la cabeza, vamos a dar un movimiento de retroceso a los cañones en el momento de disparar. Para ello, debemos crear un pequeño esqueleto formado por dos puntos. Para poder construirlo independientemente del primer esqueleto, debemos convertir el último punto de la jerarquía de éste como punto final de la cadena. Para hacer esto, seleccionamos el punto que servirá de final de cadena y abrimos las opciones de herramientas. Y en la pestaña "Joint" seleccionamos la opción "IK Chain Terminator". Para hacer efectiva la operación debemos elegir la herramienta de selección y pulsar en cualquier lugar vacío de la vista (Fig. 5).

Ya podemos crear un nuevo esqueleto en otro lugar del modelo. En la vista lateral, colocamos un punto en la base del cañón y otro en el extremo y cerramos la jerarquía con el mismo procedimiento anterior. Cambiamos a la vista superior para desplazar el nuevo esqueleto hasta situarlo dentro del cañón. Para crear el esqueleto del otro cañón, basta con copiarlo y desplazarlo. Después

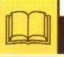


Mediante las opciones de copiado y espejo podemos crear otros esqueletos a partir de uno solo.



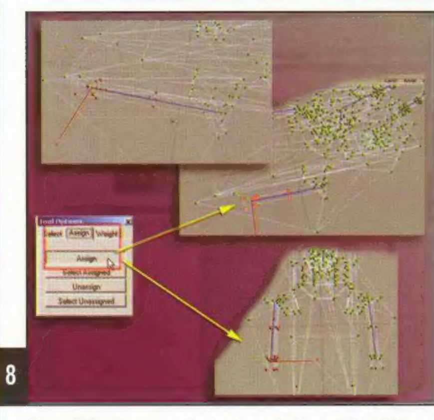
Mediante la asignación de vértices podemos asignar a un punto de unión su influencia en el modelo.

TRUCO

Podemos conmutar las operaciones de copiado y desplazamiento. Si pulsamos el botón  copiaremos y si no, desplazaremos.

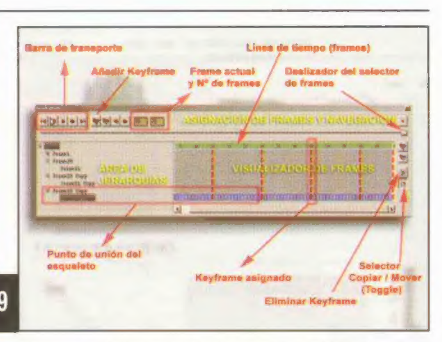
RECORDATORIO

Hay que tener en cuenta que, para animar con Character Fx, es necesario que todas las partes del modelo estén agrupadas formando un solo *mesh*.



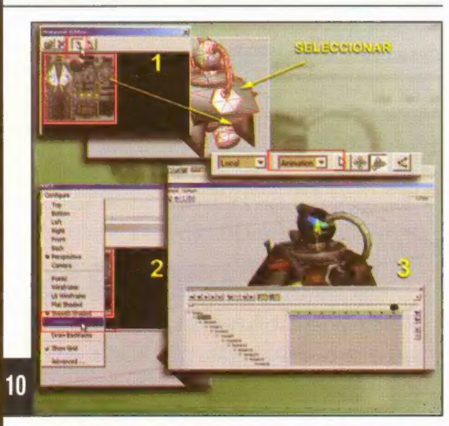
8

Es suficiente con asignar los vértices del cañón a la unión del extremo de éste para realizar un movimiento de contracción.



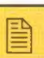
9

Esquema de las principales partes del Keyframer.



10

Los diferentes procedimientos para aplicar un material al modelo.



NOTA

En la versión comercial de Blitz3D 1.76 o superior es conveniente grabar en formato .B3D pues es muy compatible con todos los métodos de colisión.

tenemos que invertirlo horizontalmente con la opción "Mirror Objects"en el menú "Edit" (Fig. 6).

Una vez creado el esqueleto, debemos asignar a cada unión la influencia que tendrá en el modelo cuando se mueva. Esta influencia se refleja mediante los vértices.

ASIGNANDO VÉRTICES.

Pulsamos en el icono de edición de vértices. Se visualizarán todos los vértices del modelo. Empezamos por el punto principal, el del cuello. Hacemos clic sobre él para seleccionarlo, y a continuación, con mucho cuidado, vamos seleccionando todos los vértices del cuello, cabeza y todos los del tubo. Debemos realizar la operación con la tecla *CTRL* pulsada, para ir sumando selecciones. Cuando elegimos un vértice, éste se volverá de color rojo (Ver Fig. 7).

Una vez seleccionados los vértices, debemos asignarlos al punto elegido. Pulsamos con el botón derecho sobre el icono de selección de vértices. Aparecerá la ventana flotante de opciones.

Pero antes, elegimos un "peso" (influencia) de esos vértices en el modelo de 1 (por defecto) en la pestaña "Weight". Para asignar, pulsamos en el botón "Assign" de la pestaña "Assign". Observamos cómo todos los vértices cambian al color elegido en "Weight", indicando que hemos realizado la selección. Hacemos la misma operación de asignación para ambos cañones como se muestra en la figura 8.

CREANDO LA ANIMACIÓN

Ha llegado el momento de ver si realmente hemos asignado correctamente los vértices realizando el movimiento de la cabeza. Conmutamos al modo "Animation". Al hacerlo aparecerá la ventana "Keyframer" que es donde almacenaremos nuestra animación a lo largo de un determinado tiempo. Su estructura la podemos dividir en tres partes: las herramientas de asignación de frames y navegación, el área de jerarquías y el visualizador de frames. En la figura 9 se muestra una descripción detallada de este editor.

Para probar el movimiento vamos a cambiar de vista y de modo de visualizado. En el menú "Configure" de la ventana de vista elegimos "Smooth Shaded" y "Perspective". Rotamos la vista con el icono de *rotar cámara* para situar el modelo de espaldas. Seleccionaremos el icono de rotación en la paleta de herramientas y pulsamos sobre el punto de unión del cuello. Mientras pulsamos sobre la flecha amarilla (eje Y) movemos el ratón, observamos cómo la cabeza gira llevando consigo al tubo también. Una vez comprobado que todo está correcto, pasaremos a grabar el movimiento en una secuencia de frames. Para observar con más detalles la evolución del modelo durante el movimiento, sería conveniente aplicarle la textura que llevará. Elegimos el modo de visualización "Textured" y seleccionamos el modelo en el modo "Objects". Abrimos el editor de materiales en "Windows / Material Editor". En este editor, pulsamos sobre el icono "Open". Elegimos del directorio "Extras" del CD "Textura_bionave.png" y pulsamos sobre el icono de asignación (Fig. 10).



En el próximo número...

... terminaremos la animación de nuestro personaje y seguiremos creando elementos del juego.

Creando los sonidos del entorno y el menú

Una parte muy importante de nuestro juego son los elementos de audio necesarios para crear ambientación; es decir, todos los efectos de adorno que aumentan la credibilidad de lo que mostramos, como por ejemplo, el sonido de los animales, plantas, el entorno o la voz en off (voz de fondo).

¿Qué sería de un juego sin efectos de audio tales como el rugir de una bestia después de recibir un impacto de nuestro cañón o el clamor del viento en un día nevado? A continuación, vamos a dotar de sonido a los seres vivos que pululan por los terrenos de "Zone of Fighters".

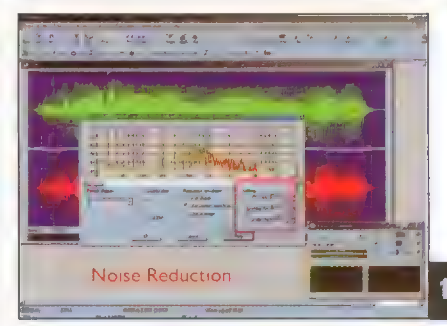
SONIDOS DEL ENTORNO

La idea básica es utilizar nuestros propios medios para conseguir efectos de audio originales. Por ello, explicaremos cómo obtener lo que queremos de una manera "artesanal". Así, para lograr dar vida sonora a los seres que pueblan el juego, utilizaremos nuestra propia voz. Evidentemente, lo que pretendemos es acercarnos lo más posible a nuestra idea, buscando y utilizando los procedimientos adecuados. De modo que nuestra calidad de voz no influirá mucho en el resultado final, pero sí lo hará nuestro trabajo en el editor de audio.

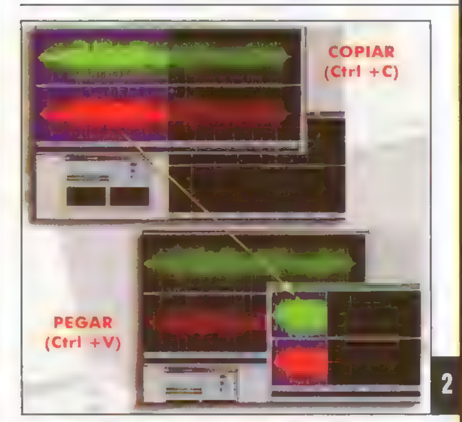
PLANTAS CARNÍVORAS Y GUSANOS

Pretendemos que cuando un *Dreeck* (planta carnívora) o un *Slunk* (gusano subterráneo) reciban el impacto de un disparo reproduzcan un rugido. Vamos a disponer, de base, de la grabación de nuestra propia voz a través de la tarjeta de sonido del ordenador. En primer lugar, y una vez dentro de nuestro editor Goldwave, debemos crear una plantilla de sonido nueva con calidad CD, a 22 Khz. y en estéreo. Recuerda que hay que trabajar de base con buena calidad y luego transformar al formato deseado. A continuación, preparamos el micrófono y asignamos el volumen adecuado para no saturar la grabación. Pulsamos el botón "Record" en "Device Controls" y pronunciamos "AAAH" en tono plano y sin subidas ni bajadas de volumen. Una vez grabado, podemos aplicar un filtro de reducción de ruidos para eliminar la posible basura existente en frecuencias altas (Fig. 1).

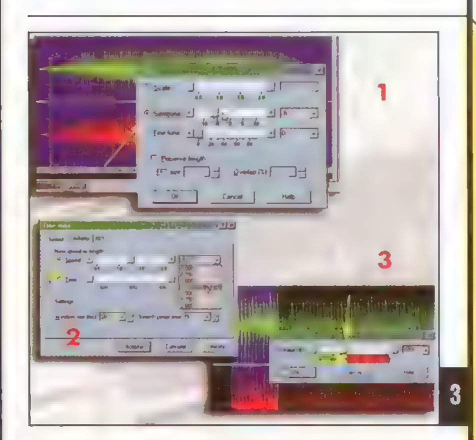
Elegimos la opción "Noise reduction" en "Filter" del menú "Effects" y aplicamos el filtro con los siguientes valores: detalle de análisis del espectro y de la envolvente del ruido (FFT size) en 12, un porcentaje para el cálculo de la envolvente (Overlap) del 80 % y una escala de actuación (Scale) del 100%. Seguimos, aplicando un normalizado, y posteriormente borramos los espacios en blanco del principio y final de la grabación. De todas formas, en el directorio "Extras" del CD se encuentra un ejemplo en formato .WAV llamado "aah.wav". Ya tenemos nuestro sonido base listo para procesar. Vamos a crear



Aplicando un filtro "noise reduction" podemos eliminar ruido en las frecuencias altas del sonido.

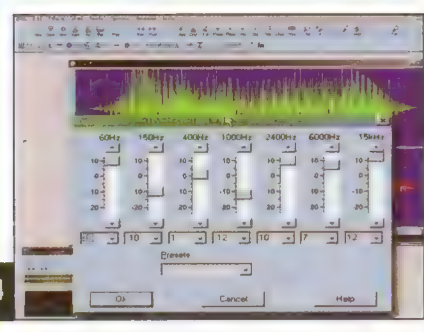


GoldWave permite realizar copias de porciones de audio a otra plantilla nueva.

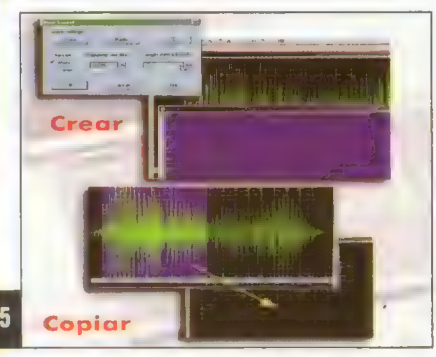


Cambiando el tono (Pitch) a un sonido podemos cambiarlo totalmente.

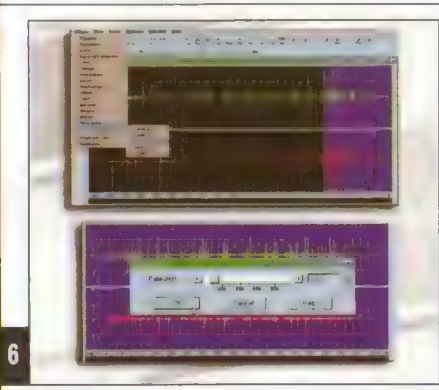




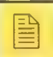
La ecualización nos ayudará a obtener un efecto con más sonoridad.



Se pueden crear plantillas de audio con distinto formato e intercambiar información de sonido entre ambas.



Por medio de la función de resampleo podemos cambiar el frame rate de un sonido.



NOTA

Para grabar la voz en off, crear una plantilla de audio de mucha duración y grabar ahí, de una forma continua, todas las frases y palabras que tendrá el juego para luego ir recortando y procesando individualmente.
Aplicamos un efecto de mecanizado eligiendo la opción "Effects \ Mechanize", designando una calidad del 20%, y normalizando el volumen después.

otra plantilla nueva vacía para ir componiendo ahí nuestro efecto de audio. El primer sonido que crearemos será el rugir lamentoso de un *Dreeck* cuando es alcanzado. Para ello, seleccionamos, más o menos la mitad del audio base. Hacemos *CTRL+ C* para copiar, y lo pegamos en la plantilla vacía nueva, situándonos en ella y pulsando *CTRL+V*. Luego, eliminamos todo el sobrante, seleccionándolo y borrando con *CTRL + X*. A partir de ahora trabajaremos en la nueva plantilla (la plantilla base la podemos minimizar) (Fig. 2).

En primer lugar, apliquemos un "Fade in" al principio y un "Fade out" al final. Seguimos, reduciendo 6 semitonos en "Effects \ Pitch" y eligiendo la opción "Time warp" en "Effects" realizaremos una deformación de la duración cambiando la velocidad en la pestaña "Similarity" a un valor de 1.250. Seleccionamos un trozo del sonido para aumentarle drásticamente el volumen con la opción "Effects \ Change Volume" a 170 (Ver Fig. 3).

Volvemos a aplicar un cambio de tono pero guardando la longitud (Preserve length) a -6 semitonos y después normalizamos de nuevo. Para lograr mayor profundidad, aplicamos una ecualización normal con los valores que se muestran en la figura 4.

Para terminar, lo grabamos con la calidad de 8 bits mono a 11025 Hz con el nombre "simpacto_dreeck". Para el rugir del *Slunk* en un impacto utilizaremos el mismo sonido creado para el *Dreeck* pero cambiándole el tono y la velocidad. Creamos una plantilla nueva, esta vez ya con la calidad final, es decir, "Voice" mono a 11025 Hz, y con una longitud de 1. Seleccionamos una parte del sonido el *Dreeck* y lo copiamos a la nueva plantilla (Fig. 5) y a continuación cambiamos de tono y velocidad hasta lograr lo deseado.

Para el *Luny* crearemos un efecto de audio que emule algo viscoso. Un buen método es grabar el roce de una bolsa de plástico grueso o de papel. Preparamos la plantilla nueva con calidad CD y grabamos. Es suficiente con volver a resamplear el sonido a menos calidad (11025 Hz) en la opción "Resample" de "Effects". Añadimos un poco de graves con el ecualizador y grabamos con calidad 8 bits, mono (Fig. 6).

SONIDO DEL MENÚ

En el menú del juego sólo se oye la música de fondo y un pitido le acompaña cada vez que se acepta un grupo de opciones. Esta especie de clic nos servirá de referencia para la opción del menú de cambio de volumen de efectos. Para realizar este pitido, vamos ha utilizar el evaluador de expresiones de Goldwave. Creamos una plantilla nueva con calidad 8 bits, 11025 Hz y mono (Óbice) con una duración de 1. Entramos en el evaluador en "Tools \ Expression evaluator". Utilizaremos una onda sinusoide: escribimos la función *sin(2*pi*f*t)* donde el tiempo (t) vale 0.32 y la frecuencia (f), por ejemplo, 200 (a mayor cantidad más agudo sonará el pitido). Pulsamos en el botón "Start" y, seguidamente, en "Device Controls" (no hace falta cerrar el evaluador de expresiones) pulsamos en "Play" para oír el resultado. Para terminar, nos quedamos sólo con un trozo del audio y salvamos en disco.

En el próximo número...

... empezaremos a estudiar las herramientas necesarias para la realización de la música de nuestro juego.

Creación y manejo de terrenos. Mundos BSP

Antes de empezar con el tratamiento de terrenos queda pendiente el estudio de algunas funciones importantísimas para lograr implementar un sistema de colisiones completo.

FUNCIONES ESPECIALES PARA CONTROLAR COLISIONES

B3D nos proporciona otras funciones muy útiles y necesarias para implementar un buen control de colisiones en nuestros juegos. En ocasiones necesitaremos saber exactamente con qué entidad ha chocado nuestro disparo de entre muchas del mismo tipo. Debemos entonces realizar un procedimiento en el que se ven involucradas algunas funciones muy específicas. Por ejemplo, estamos disparando a un grupo de ovnis, todos del mismo tipo de colisión: ENTIDAD _OVNI. Y queremos saber a qué ovni del grupo se ha alcanzado. Debemos seguir los siguientes pasos:

- **1.** Preguntar si el disparo ha colisionado con un OVNI:

```
If EntityCollided ( [illegible]
[illegible] ) = True
```

- **2.** Recorrer en un bucle todas las colisiones detectadas en un ciclo de "UpdateWorld" con la función *CountCollisions* (entidad):

```
For [illegible] = [illegible] To CountCollisions
( [illegible] )
```

- **3.** Preguntar a cada una de las colisiones detectadas si se trata de una colisión con un tipo ovni. Para ello debemos primero obtener el tipo de colisión de la entidad con la función:

```
GetEntityType ( [illegible] )
```

Y, a continuación, saber cuál es la otra "entidad" involucrada en la colisión con la función:

```
CollisionEntity ( [illegible] )
```

Donde el parámetro "índice" se refiere al número de la colisión contada con "CountCollisions".

```
[illegible]
[illegible]Entity ( [illegible] )
If GetEntityType ( [illegible]
[illegible] ) = [illegible]
...
```

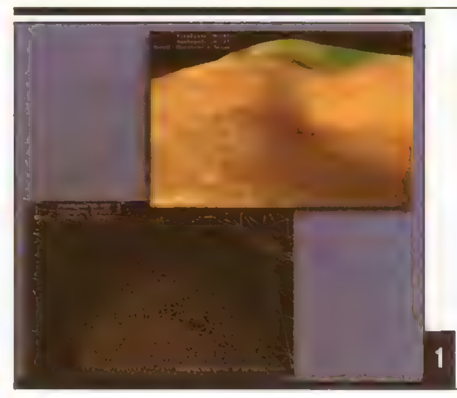
Este procedimiento parece complicado, pero tiene una lógica bastante sencilla. Simplemente, debemos preguntar al Blitz3D las mismas cuestiones que nos hacemos nosotros. Para cada cuestión, obtenemos una respuesta con alguna función. Por ejemplo, una vez que hayamos aislado la "entidad destino" involucrada en la colisión, necesitaremos obtener algunos datos más, como las coordenadas exactas de la colisión o la componente de las normales de dicha colisión, es decir, el lugar y la dirección, por ejemplo, donde recibió el impacto el ovni. Un ejemplo más claro lo encontramos cuando queremos orientar un sprite con la textura de una mancha para simular un impacto en una superficie irregular. Para obtener todos estos datos disponemos de las funciones:

```
CollisionX ( [illegible] )
```

Obtenemos la coordenada X la colisión.

```
CollisionY ( [illegible] )
```

Obtenemos la coordenada Y la colisión.



La técnica LOD es bien visible en el modo wireframe (imagen inferior).

```
CollisionZ ( [illegible] )
```

Obtenemos la coordenada Z de la colisión.

```
CollisionNX ( [illegible] )
```

Obtenemos la componente X de la normal de la colisión.

```
CollisionNY ( [illegible] )
```

Obtenemos la componente Y de la normal de la colisión.

```
CollisionNZ ( [illegible] )
```

Obtenemos la componente Z de la normal de la colisión.

Para terminar disponemos de una función que nos permite convertir una entidad en "pikeable", es decir, que puede ser detectada por otra entidad en la distancia. Nos referimos a:

```
EntityPickMode entidad, tipo
de geometría para la detección
```

El "tipo de geometría" funciona de la misma manera que las funciones de colisión:

- **0.** Unpickeable (por defecto)
- **1.** Esfera
- **2.** Polígono
- **3.** Cubo



La calidad visual del terreno dependerá también del tamaño de la textura utilizada.

INTRODUCCIÓN A LOS TERRENOS

Blitz3D permite manipular un tipo de objeto 3D formado por multitud de triángulos para la creación de paisajes. Generalmente, estos objetos están formados por cientos de miles de polígonos para poder obtener un resultado adecuado. Sin embargo, dibujar todos estos polígonos conlleva un gran esfuerzo por parte del sistema. Por este motivo, Blitz3D utiliza una técnica denominada LOD, que consiste en cambiar el detalle dinámicamente; de esta manera, se puede mantener una forma aproximada del terreno con menos cantidad de polígonos. (Ver "ejemplo1.bb") (Fig. 1)(/*insertar Blitz3D9_1.jpg) Los terrenos manejados por Blitz3D tienen ciertas características que es necesario tener en cuenta.

En primer lugar, el terreno tiene forma cuadrada y con un tamaño múltiplo de 2, por ejemplo, 2,4 .. 16 .. 512, etc.. Este cuadrado está formado por una malla cuadriculada proporcionalmente y con el mismo número de vértices en cada cuadrícula. Y por último, sólo es posible modificar la altura de un vértice cada vez.

NOTA

En una imagen para un mapa de alturas, las tonalidades negras representan las partes más bajas del terreno y las blancas las más altas.

¿CÓMO CREAR TERRENOS?

Para la creación de terrenos disponemos de dos métodos: utilizando un mapa de alturas o bien por medio de una malla de terreno simple, subiendo o bajando manualmente los vértices.

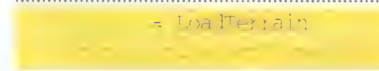
CARGANDO MAPAS DE ALTURA

Este método es el más eficaz y fácil de utilizar. Consiste en transformar un bitmap en escala de grises en una malla con alturas; es decir, a partir de una malla se generan las alturas dependiendo del tono de gris de la imagen. Es importante saber que, para que se genere correctamente el terreno, la imagen con el mapa de alturas debe ser de proporciones múltiplo de 2, como si de una textura se tratara; es decir, 2x2, 64x64 o 512x512, etc.. (Fig 2.)

A continuación vamos a realizar todos los pasos para crear un terreno a partir de un mapa de alturas.

En el "ejemplo2.bb" se muestra todos estos pasos de una forma práctica (Fig. 3).

En primer lugar debemos cargar la imagen y crear el terreno con la función "LoadTerrain".



Aunque parezca mentira, esto es básicamente todo lo que hay que hacer.

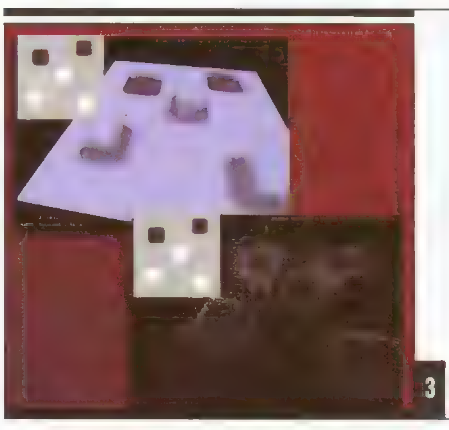
Blitz3D crea el terreno con las dimensiones de la imagen; es decir, si el mapa de alturas tiene 512 x 512 de lado, el terreno se creará con 512 píxels de lado. Sin embargo, la altura será siempre 1.

Para poder cambiar su tamaño, necesitaremos utilizar la función "ScaleEntity".

Por ejemplo: queremos tener un terreno de 15.360 píxels de lado; es decir, en el eje X y en el eje Z, y una altura de 350 con un mapa de 512 x 512. En este caso, escribiremos:

```
= LoadTerrain

ScaleEntity Terreno, 30, 350,
30  ➨ 30 x 512 = 15.360
```



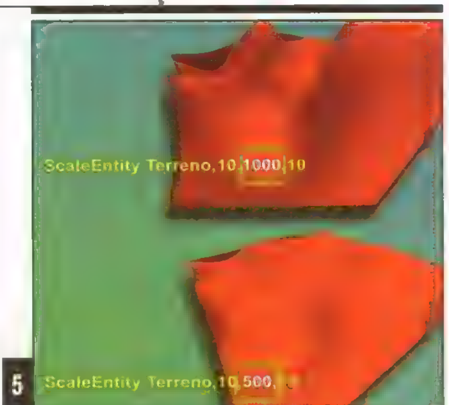
Un ejemplo de creación de terrenos utilizando mapas de alturas. Los colores claros representan elevaciones en el terreno y las oscuras hondonadas.

Hay ciertos factores que influyen en la generación de un buen terreno. El principal de todos es cómo dibujemos el mapa de altura. Unos cambios de tonalidad bruscos pueden ocasionar un terreno con filos y cortes, sin embargo, si utilizamos transiciones suaves utilizando desenfoques o *blur* conseguiremos alturas suavizadas (Ver Fig. 4).

Otro factor a tener en cuenta es la altura que le asignemos, escalando la coordenada, ya que a mayores alturas los polígonos



Dependiendo de la suavidad en las tonalidades de color en el mapa de alturas lograremos un terreno más o menos redondeado.



5

El escalado del terreno en su eje Y provoca un mayor porcentaje en la representación de las elevaciones.

se estirarán más, dando peor resultado. En la figura 5 se puede observar la diferencia entre dos diferentes alturas dadas a un mismo terreno.

A su vez, hay que tener en cuenta las proporciones en los ejes X y Z a la hora de aportar un valor al eje de altura Y.

CREANDO UN TERRENO A PARTIR DE UNA MALLA

Además de utilizar un mapa de alturas es posible crear un terreno a partir de cero con la función

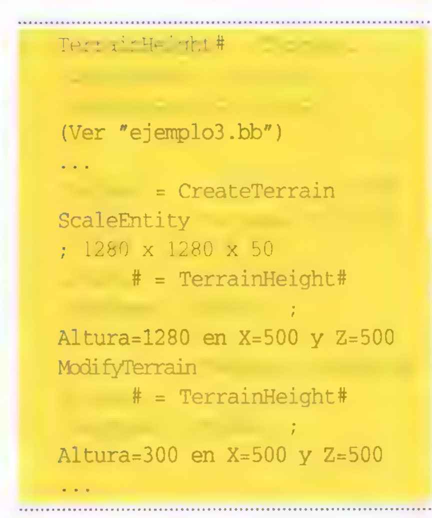
```
"CreateTerrain": Terreno =
CreateTerrain
```

Con esta función crearemos un terreno del tamaño que indiquemos en "tamaño del grid", por ejemplo en: *Terreno = CreateTerrain (512)* Estamos creando un terreno de 512 cuadrículas y de tamaño 512 x 512 x 1 de altura. Hay que tener en cuenta que el tamaño del grid también debe ser siempre múltiplo de dos. Una vez que hemos creado la malla debemos levantar o bajar los vértices para crear las alturas. Debemos recordar que sólo es posible cambiar la altura de un vértice cada vez. Utilizaremos para ello la función *"ModifyTerrain"*:

```
ModifyTerrain
```

En el "ejemplo3.bb" vemos cómo funciona este procedimiento (Fig. 6).

Es muy útil poder saber en cualquier momento qué altura hay en cualquier coordenada del terreno. Para obtener esta información disponemos de la función *"TerrainHeight"*:



```
TerrainHeight#

(Ver "ejemplo3.bb")
...
 = CreateTerrain
ScaleEntity
; 1280 x 1280 x 50
# = TerrainHeight#
;
Altura=1280 en X=500 y Z=500
ModifyTerrain
# = TerrainHeight#
;
Altura=300 en X=500 y Z=500
...
```

CONTROLANDO EL NIVEL DE DETALLE

Mediante la función "TerrainDetail" podemos controlar el número de polígonos que Blitz3D utilizará para generar el terreno.

```
TerrainDetail
```

Sabemos que Blitz3D utiliza el sistema LOD o "nivel de detalle" para reducir los polígonos del terreno situados a gran distancia de la cámara. Esta técnica, a veces, produce la sensación de que el terreno se mueve

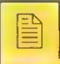

6

A partir de una malla podemos construir un terreno subiendo o bajando cada vértice.

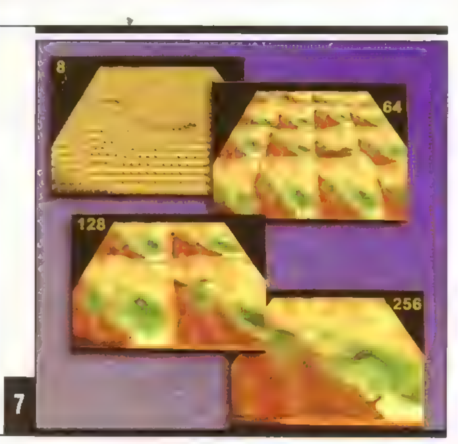
(sobre todo cuando se utilizan pocos polígonos). Para disimular este inconveniente visual es recomendable activar la opción "transformación de vértices" con "True".

El número de polígonos está limitado a la potencia del ordenador y sobre todo a la memoria de vídeo disponible (tarjeta gráfica). Evidentemente, la calidad de la representación del terreno, generalmente a partir de mapas de alturas, depende de la cantidad de triángulos que utilicemos. Sin embargo, Blitz3D utiliza una técnica que permite adaptar lo más posible el número de polígonos a la forma original del terreno. Es bueno saber algunos conceptos para averiguar, en cierta medida, la cantidad de polígonos que se necesitarán para obtener un resultado fiel. Por ejemplo, las necesidades poligonales aumentarían si el terreno tuviese demasiadas variaciones bruscas de alturas o si lo escalamos demasiado. También es importante aplicar técnicas para reducir la demanda de polígonos, como por ejemplo: suavizar los bordes en el mapa de altura, reducir el rango de visualización de la cámara o escalar el terreno más allá del límite de visión de la cámara.

También podemos aplicar al terreno sombreado ("shading"), el cual puede cambiar si utilizamos y variamos una luz direccional: *TerrainShading Terreno, True/False*.


NOTA

Se pueden obtener mapas de alturas a través de aplicaciones especializadas como Bryce o bien dibujadas directamente con un programa de dibujo como Paint Shop Pro.



Mediante el escalado de la textura controlamos el tileado.

Si usamos este efecto, el rendimiento se reduce, ya que Blitz3D necesita calcular los polígonos afectados por la sombra. Además, visualmente se notaría más el sistema LOD aplicado al terreno.

TEXTURIZANDO TERRENOS

Un terreno se crea con un solo color plano. Así que mediante la función "EntityTexture" podemos asignarle texturas, por ejemplo, para crear el aspecto de la tierra.

El inconveniente radica en que no podemos texturizar partes diferentes del mismo terreno, ya que Blitz3D lo trata como una sola entidad, precisamente por este motivo, la textura lo cubre por completo utilizando la técnica *Mip Mapping*. En el ejemplo:

```
          = LoadTexture

          = CreateTerrain
EntityTexture
```

La imagen "tierra.bmp" se aplicará a cada uno de los 128 cuadros que forman la cuadrícula del terreno, formando un mosaico (tileado) de la textura (Fig. 7).

Esto es debido a que, por defecto, se le asigna un tamaño de 1, el cual corresponde a un cuadrado del mallado. Si queremos que sólo una imagen cubra el terreno es preciso escalar la textura al tamaño de éste antes de aplicarla, es decir:

```
ScaleTexture
```

Se pueden obtener resultados realmente brillantes mediante el multitexturizado; es decir, la aplicación de más de una textura al terreno (Ver ejemplo9_4.bb) (Fig. 8).

Es posible aplicar hasta 8 texturas; sin embargo, esta técnica reduciría el rendimiento. Además, esta posibilidad depende de la tarjeta gráfica, ya que no todas soportan tantas mezclas de texturas en el mismo objeto. Generalmente, la posibilidad común es de dos o tres e incluso cuatro en tarjetas de alto nivel como la *GeForce 3* (Fig. 9).

```
          = LoadTexture
          = LoadTexture
          = CreateTerrain
ScaleTexture
EntityTexture Terreno,Textura1,0,0 ;
0, 0  frame de la textura y
número de textura

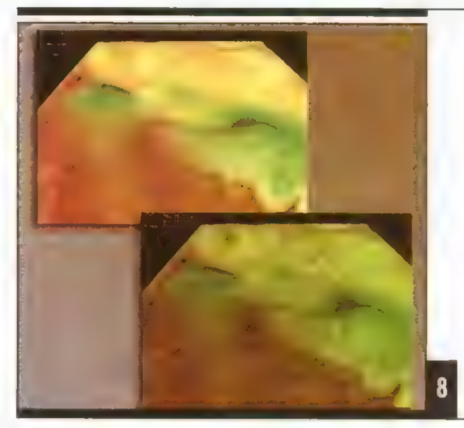
EntityTexture Terreno,Textura2,0,1 ;
0,1 frame 0 y textura número 2
```

Cabe resaltar que es posible emplear diferentes tamaños a cada una de las texturas aplicadas al terreno.

Puede ocurrir que se esté ejecutando una aplicación o un juego que utilice esta técnica de multitexturizado para mezclar tres texturas en un objeto en equipos cuya tarjeta gráfica no soporta más de dos. En este caso Blitz3D emulará la tercera textura, creando una copia del objeto con esa textura. Evidentemente, esto reducirá en gran medida el rendimiento del programa.

OBTENIENDO INFORMACIÓN

En ocasiones podríamos necesitar algunos datos referentes al terreno, como por ejemplo su tamaño o sus coordenadas. Para ello, disponemos de las funciones:



Mediante el blending o mezcla de texturas se pueden lograr resultados más realistas.

TerrainSize (Terreno) Retorna el tamaño de la cuadrícula del terreno.

Con las funciones "TerrainX", "TerrainY", "TerrainZ" podemos obtener la posición exacta dentro del terreno.

```
TerrainX#
TerrainY#
TerrainZ#
```

Si, por ejemplo, andamos por el terreno y queremos saber exactamente a qué altura del terreno estamos utilizaríamos "TerrainY".



Blitz3D permite mezclar hasta 8 texturas a la vez, sin embargo esta posibilidad depende de la capacidad de la tarjeta de vídeo. Lo normal es mezclar dos o incluso 3 en tarjetas más potentes.



En el próximo número...

... empezaremos con las funciones 3D. Estudiaremos detenidamente el manejo de objetos en 3D.

Editores de audio. Cool Edit Pro 2.0 (I)

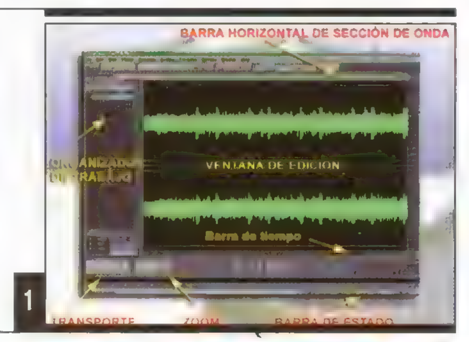
Continuamos la serie de tutoriales dedicados a los editores de audio con la segunda versión de Cool Edit Pro de Syntrillium Software.

Este programa tiene la posibilidad de grabar y mezclar hasta 128 pistas de audio en estéreo mediante cualquier tarjeta de sonido compatible con Windows. Pero antes de centrarnos en esta estupenda cualidad, vamos a tener una primera toma de contacto con su interfaz de usuario y aprenderemos a navegar por la ventana de edición y las funciones más importantes.

Al igual que ocurría con Sound Forge, Cool Edit permite el sincronismo de audio con vídeo y además posee más de 40 efectos y procesos de audio en tiempo real, así como herramientas para la restauración de audio en malas condiciones. Otras características son la posibilidad de generar tonos y ruidos, que soporta calidades de hasta 32 bits de resolución y que permite la creación de loops.

PRIMERA TOMA DE CONTACTO

Cool Edit Pro funciona con dos espacios de trabajos importantes: la ventana de edición y la ventana multipista.



Descripción de las principales partes de la ventana de edición.

VENTANA DE EDICIÓN

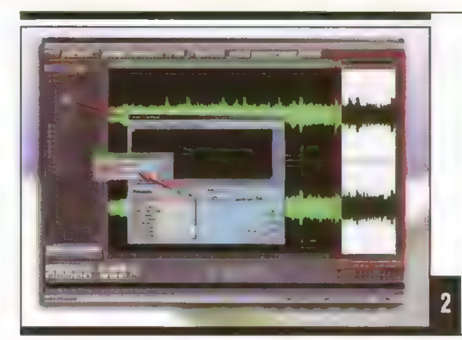
Lo primero que notamos es que la disposición de las ventanas e iconos es más densa que en Sound Forge. Distinguimos una ventana opcional y configurable en la parte izquierda, en la que se encuentra un organizador del trabajo. Podemos ocultarla o visualizarla con *ALT + 9*. En ella podemos tener la lista de todos los sonidos que carguemos, así como los efectos disponibles. El espacio restante lo ocupa la ventana de ondas donde se representa el sonido y donde lo editaremos. En la figura 1 se muestra una descripción de esta ventana.

VENTANA MULTIPISTA

Pulsando en el icono accedemos a la sección multipista del programa. En este espacio de trabajo unimos y mezclamos diferentes sonidos a través de pistas, al igual que lo haríamos con cualquier secuenciador MIDI. Además, podemos aplicar efectos en tiempo real a cada una de las pistas que editemos.

CARGAR Y EDITAR SONIDOS EN LA VENTANA DE EDICIÓN

Los procedimientos de carga y edición son muy similares a otros editores de audio, como por ejemplo Sound Forge 6.0. Vamos a realizar algunas operaciones básicas que ya hemos utilizado en ocasiones anteriores para conocer cómo trabaja Cool Edit. Empezamos cargando un sonido contenido en el CD, en el directorio "Extras", llamado "loop1.wav" Esta operación la podemos realizar pulsando en cualquiera de los iconos . Una vez cargado, el siguiente paso será reproducirlo, pulsando en el icono o con la tecla *ESPACIO*. A continuación,



Procedimiento para sumar a la librería un efecto propio. En este caso un FADE.

vamos a aplicar un "Fade in" al principio y un "Fade out" al final. Antes de seguir debemos seleccionar con el ratón la parte final del sonido a la que le aplicaremos el efecto. En Cool Edit no disponemos exactamente de esa función, así que la tendremos que fabricar nosotros. En realidad, un cambio del volumen significa un cambio en la envolvente del sonido, por lo que elegimos el efecto "Envelope" en "Effects \ Amplitude \ Envelope" del menú, o bien seleccionándolo a través del organizador de trabajo, pulsando en la pestaña "effects" y abriendo los menús hasta encontrar "Envelope". Al pulsar dos veces, aparece la ventana "Create Envelope" en la cual se muestra una gráfica y una lista de "Presets". Nuestra intención es sumar a la lista (librería) nuestro Fade Out. Para ello situamos la gráfica como se muestra en la figura 2, que representa una caída del volumen hacia 0.

Una vez dibujada la gráfica pulsamos en el botón "Add" y escribimos el nombre del efecto "Fade Out" y "Ok". Podemos realizar un previo del efecto pulsando en "Preview". El "Fade in" lo haremos de la misma forma pero seleccionando el principio del sonido. Lo siguiente que haremos será, como siempre, normalizar el sonido con



3

Procedimiento para crear listas de selecciones.

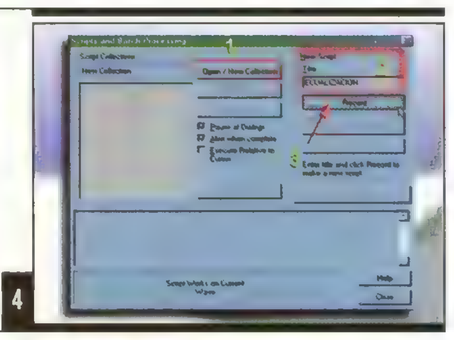
"Normalize" a un 80%. A diferencia de lo que ocurría con Sound Forge, en Cool Edit no podemos tener a la misma vez más de una ventana abierta, aunque sí varios sonidos cargados a la vez con la posibilidad de conmutar entre uno y otro en el menú "Window".

También encontramos aquí la posibilidad de tener una lista de selecciones. Para insertar en la lista es suficiente con crear una selección nueva y pulsar *F8*, pero antes podemos ver la lista pulsando *ALT + 8*. En la figura 3 se muestran los pasos para editar la lista de selecciones.

Si deseamos saber cómo sonará el sonido con otra frecuencia (pero sin llegar a transformarlo) utilizaremos la opción "Adjust Sample Rate" en "Edit"; pero si de verdad queremos convertirlo a otro formato utilizaremos "Convert Sample Rate" en "Edit" o pulsando *F11*.

USANDO SCRIPTS

Una buena utilidad que proporciona Cool Edit Pro es la posibilidad de crear scripts para la grabación de operaciones. Esta opción permitirá almacenar en un fichero de texto los efectos y operaciones exactas que se aplican a un sonido para poder, en un futuro, asignarlas idénticamente a otro sonido diferente. Abrimos la ventana "Scripts and Batch Processing" en "Options". En primer lugar debemos crear un script nuevo y darle un nombre al archivo, por ejemplo "SCRIPT1", y pulsando el botón "Open / New Collection". Una vez creado el nuevo archivo, añadiremos operaciones a la nueva colección de acciones. Vamos a aplicar una ecualización al sonido. Nos situamos en "Title" en "New Script"y escribimos el nombre de la operación, por ejemplo, "ECUALIZACION". Se activará el botón "Record" (Fig. 4).



4

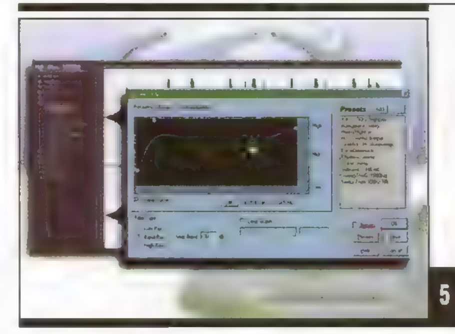
Antes de fabricar cualquier script es necesario crear el fichero que lo contenga.

Una vez pulsado, la ventana se cerrará y el programa quedará a la espera de que se realice la operación de ecualización. Elegimos, por ejemplo, el efecto "Dynamic EQ" en "Filtres". Dibujamos la gráfica como se muestra en la figura 5.

Al pulsar "Ok" el efecto se aplicará. Volvemos a la ventana "Scripts and Batch Processing" y pulsamos el botón "Stop Current Script" para que el programa termine de grabar acciones. Para registrar definitivamente el nuevo script en la colección pulsamos en el botón "<<Add to Collection>>". Podemos editar el nuevo script a través del texto con la configuración generada pulsando en "Edit Script File" (Fig. 6).

Añadamos otro nuevo script a la colección. En "Title" escribimos "TONO" y pulsamos en "Record". Elegimos el efecto "Pitch Bender" en "Time/Pitch" en el menú "Effects". Modificamos el transcurso del tono en el sonido y aceptamos.

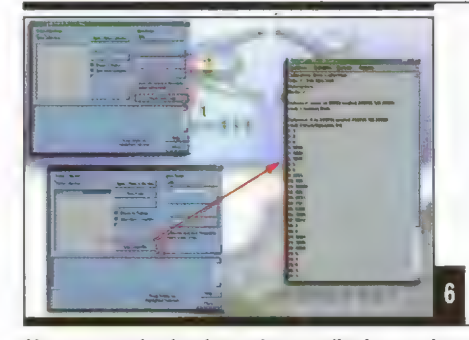
Abrimos la ventana de scripts y realizamos el mismo proceso. Ahora tenemos en disco un archivo llamado "SCRIPT1" que contiene los procesos de ecualización y cambio de tono, los cuales podemos aplicar de la misma forma a otro sonido diferente. Cerramos el sonido que tenemos con *CTRL + W* y cargamos uno nuevo o el mismo. A continuación, entramos en la ventana de scripts y abrimos "SCRIPT1" pulsando el botón "Open / New Collection". Ahora podemos ejecutar cada uno de los scripts de la lista seleccionándolo y pulsando en "Run Script". Se abrirá la ventana del efecto seleccionado, por ejemplo "ecualización", con los mismos parámetros aplicados anteriormente.



5

La ecualización dinámica nos permite dibujar de una forma flexible el ecualizado de nuestro sonido.

Es inevitable observar que el funcionamiento de esta aplicación, sobre todo en los procesos de edición y aplicación de efectos, se asemeja mucho al manejo de Sound Forge 6.0. Sin embargo, el sistema multipista es exclusivo de Cool Edit Pro 2.0 y una herramienta potentísima para la creación de audio. Precisamente explicaremos su funcionamiento, de una forma práctica, realizando la mezcla que hizo posible el audio argumental de "Zone of Fighters".



6

Una vez creado el script podemos editarlo con el bloc de notas de Windows.



En el próximo número...

... aprenderemos las posibilidades del multipista de Cool Edit Pro 2.0

Arcades. Acción/Aventura 3D y subgéneros tácticos

Con la llegada del juego *Lara Croft: Tomb Raider* (Fig. 1), los Arcades 3D adquieren un nuevo concepto: la aventura.

Esta nueva forma de juego permitía al jugador interrelacionarse aún más con la historia, pudiendo, a veces, cambiar su curso dependiendo de su actuación durante el juego. De esta manera, el desarrollo de la partida se hace cada vez menos lineal y aburrido, ya que el jugador necesita resolver, para poder avanzar, ciertos problemas en forma de puzzles, combinaciones de botones o palancas, o por medio de objetos secretos como llaves o claves de acceso. El mundo 3D que el jugador tiene delante se expande en posibilidades, permitiendo una exploración libre del entorno en busca de respuestas.

LA CLAVE DEL ÉXITO, UNA MUJER

Sí, fue el hecho de que la protagonista de Tomb Raider fuera una mujer lo que, en parte, generó el éxito de este juego; además, claro está, del nuevo concepto que introdujo en el género de los Arcades 3D. Cuando, diseñado por Core Design Ltd., fue lanzado en 1996 por Eidos Interactive, sólo se conocían héroes como *Mario* de Nintendo, el "machote" de *Duke Nukem* o el brazo del guerrero de *Doom* o *Quake*. El hecho de transportar la cámara a una tercera persona posibilitó que el jugador adoptara el rol del protagonista del juego. El personaje de la saga *Tomb Raider* tenía nombre propio, *Lara Croft*, e incluso una vida propia.



DEFINICIÓN

▸ VOXEL

Voxel significa crear superficies y modelos 3D por medio de volúmenes creados por píxels. Los voxels pueden tener el tamaño de un punto de pantalla, de una esfera o de un sprite.

UN MUNDO DE AVENTURAS COMIENZA

El nuevo concepto de aventura aumentaba, aún más, el número de acciones que el jugador podía realizar. Además, se respiraba un cierto aire de Rol en los juegos. Pronto, el nuevo estilo impuesto por *Tomb Raider* inundó las cabezas pensantes de las desarrolladoras que vieron cómo los nuevos Arcades 3D interactivos se hacían dueños del PC. Muchos títulos siguieron estas pautas, como *Outcast* (Infogrames, 1999). Aunque habían pasado algunos años desde el lanzamiento del primer *Tomb Raider, OutCast* (Fig. 2) significó un avance técnico importante en los juegos de acción y aventuras 3D, ya que poseía una interactividad absoluta con el entorno y los personajes gracias a la mejor IA conocida hasta el momento.

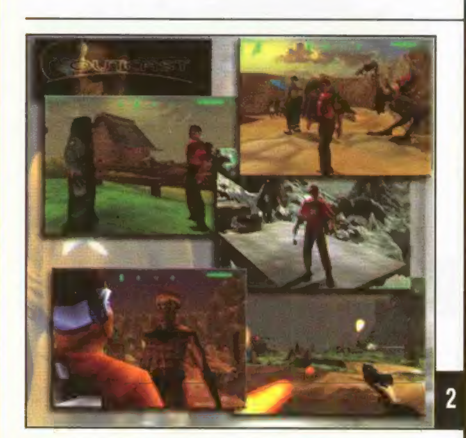
Además, su motor 3D poseía cualidades gráficas envidiables, como el renderizado de polígonos y voxel, texturas en tiempo real con bump-mapping, sombras y luces dinámicas, animaciones por "Motion Capture" y técnicas cinematográficas para el manejo de las cámaras.

A partir del año 1999 la mayoría de las producciones de acción 3D rondaban en torno a la aventura como *Episodio I: La Amenaza Fantasma* (Lucas Arts, 1999), *Prince of Persia 3D* o el original *Messiah* (2000) (Fig. 3).

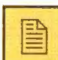
La proliferación del sistema multijugador como reclamo co-



La llegada de Lara Croft: Tomb Raider creó el concepto de aventura en los arcades 3D.



OutCast llevó la aventura a los arcades 3D de una manera única. Permitió al jugador toda la libertad para interactuar con el entorno y los personajes.

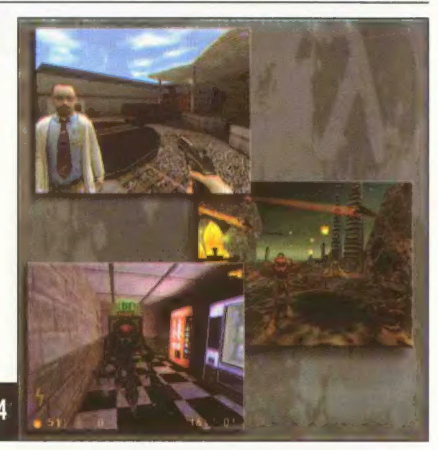


NOTA

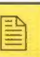
Aunque la acción 3D de los FPS clásicos cada vez quedaba más a un lado frente a la acción-aventura, aparecieron juegos que, por su calidad gráfica y jugabilidad se han convertidos en clásicos del género como: *Jedi Knight: Dark Forces II* (Lucas Arts, 1997), Unreal (Epic Games, 1998), *Thief: The Dark Project* (Eidos Interactive, 1998), *System Shock II* (Electronic Arts, 1999) o *MDK 2* (Shiny, 2000)


3

Dos ejemplos de auténtica aventura en un juego de acción 3D: Episodio I: La Amenaza Fantasma y Blade: The Edge of Darkness. Este último con posibilidades multijugador para combates DeathMatch (arena).


4

Half Life cambió completamente el concepto de los FPS clásicos. Su concepción poco a poco introdujo el subgénero táctico en los arcades 3D.



NOTA

La inteligencia artificial (IA) de *Outcast* utilizaba el sistema GAIA, el cual modificaba el comportamiento de los personajes a través de agentes. Estos agentes lo formaban la misma acción del juego y eran captados por los personajes, que actuaban según las circunstancias. Se simuló un universo real donde todo estaba relacionado y donde la fauna incluso necesitaba comer, beber o dormir.

mercial importantísimo propició que sistemas de juegos diferentes se unieran en un mismo producto. Es el caso de juegos como *Blade: The Edge of Darkness* (Rebel Act Studio, 2001) o *Rune* (con un sistema acción-aventura para un jugador y un modo arena de combate para multijugador (Fig. 3).

NUEVOS SUBGÉNEROS: FPS Y ARCADES EN TERCERA PERSONA TÁCTICOS

Sin duda, el concepto básico de los FPS se vino abajo con la incursión en el PC del juego *HalfLife* (Sierra, 1998) Este título olvidó los conceptos clásicos en este tipo de juegos, como la activación de palancas o la búsqueda de códigos de acceso para ir resolviendo los niveles. En su lugar, confirió al jugador la tarea de involucrarse en las situaciones como en la vida real, interrelacionándose con los personajes y encauzando los problemas de una manera lógica. Encontramos entonces niveles estudiados al milímetro con puzzles, enemigos inteligentísimos y detalles excepcionales. Y todo desde una perspectiva subjetiva para no olvidar lo que realmente gustaba al jugador del género (Fig. 4).

Este nuevo concepto en los FPS avanzó, y pronto empezaron a salir extensiones de *HalfLife*, como *Counter Strike* (Rookie Studio, 2000) orientado al modo multijugador en el que el jugador forma parte de un equipo de otros jugadores en un fin común, por ejemplo, formar un grupo de policías en busca de terroristas, etc. La calidad gráfica de estos sistemas multijugador no es tan importante, ya que lo que se busca es velocidad, debido al juego masivo a través de internet, y realismo en las acciones y situaciones. Inicialmente, el concepto de táctica de grupo fue introducido en los juegos de acción 3D por la saga *Tom Clancy's, Rainbow Six* (Red Storm Entertainment, 1998) Pronto, otros títulos adaptaron estos sistemas de juegos como: *Team Factor* (7FX, 2002), la serie *Swat* (Sierra, 2000) o la competencia directa de *Counter Strike, Global Operations* (Barking Dog Studios, 2001), el cual utiliza el motor *LithTech 2.5* de indudable calidad.

Paralelamente, y debido a la gran influencia del género de estrategia en tiempo real que tanto éxito tenía en el mercado, algunos juegos de acción 3D empezaron a mezclar la acción con la estrategia de una forma magistral, como en el caso de la serie *BattleZone* (Activision, 1998) y *BatlleZone II* (1999).

Los combates cuerpo a cuerpo se van sustituyendo por sistemas tácticos organizados por el propio jugador. Ahora, el control se hacía, no de un sólo personaje, sino de todo un grupo de ellos. Este nuevo género táctico también se introdujo con mucho éxito en los arcades 3D con perspectiva en tercera persona y remontándonos a los comienzos debemos hablar sin duda del padre de todos, *Metal Gear Solid* (Konami, 2000). Este título en realidad apareció para la consola PlayStation de Sony. Se trataba de un juego que mantenía el funcionamiento de la escuela impuesta por *Tomb Raider*, es decir, un solo personaje, 3D en tercera persona y sin multijugador. Sin embargo, añadió un nuevo concepto a este tipo de género: la estrategia táctica. Un complejo sistema de seguimiento de la acción por medio de la cámara, un comportamiento totalmente real de los enemigos y unas acciones determinadas para cumplir la misión, llevaron la chispa de la estrategia a los arcades 3D.

No tardaron en aparecer nuevos títulos con este tipo de planteamientos, pero en esta ocasión se basaron en las posibilidades multijugador.



En el próximo número...

... comenzaremos una serie dedicada a la historia de los juegos de estrategias en todas sus vertientes. Empezaremos por la estrategia por turnos.

Cuestionario Videojuegos



Preguntas

1. Disponemos de un mapa de alturas de 256 x 256. ¿Cómo podemos obtener un terreno de 25.600 x 25.600 píxels y 300 de altura?

2. Enumera los pasos para texturizar mediante *blending* un terreno.

3. Escribe una función que devuelva la raíz cuadrada de un número en Blitz3D.

4. Enumera los pasos necesarios para visualizar correctamente un entorno 3D.

5. Enumera los pasos para crear una animación en Character FX.

6. ¿Cómo y dónde deben ir los diferentes movimientos de un personaje en Character FX?

7. En GoldWave hemos grabado un sonido a través del micrófono. Al oírlo notamos algún ruido. ¿Cómo podemos solucionarlo?

8. ¿Qué procedimiento debemos seguir para cambiar la calidad de un sonido en GoldWave?

9. En Cool Edit Pro 2, ¿cómo podemos fabricar un cambio de volumen en un sonido?

10. Enumera los pasos para crear un script en Cool Edit Pro 2.

Respuestas al cuestionario 8

▷ **1.** Para mover:
MoveEntity Entidad, X#,Y#,Z#
TranslateEntity Entidad, X#,Y#,Z#
PositionEntity Entidad, X#,Y#,Z#
Para rotar:
RotateEntity Entidad, Pitch(X)#,Yaw(Y)#,Roll(Z)#
TurnEntity Entidad, Pitch(X)#,Yaw(Y)#,Roll(Z)#

▷ **2.** Definimos el tipo de entidad con un número:
EntityType Entidad, Tipo_entidad (un número)
Activar el sistema de colisiones y definir la acción en caso de colisión:
Collisions Entidad_Origen, Entidad_Destino, Método_detección, Tipo_acción

▷ **3.** – Se organiza y estructura todo el código.
– Se facilita el depurado de errores.
– Permite que varios programadores puedan trabajar en el mismo programa.

▷ **4.** Porque, generalmente, son variables compartidas entre varias funciones.

▷ **5.** Antes de poder pintar el modelo en Deep Paint 3D es necesario crear un Mapeado UV para que los vértices de los polígonos del modelo se adapten al dibujo de la textura.

▷ **6.** Podemos pintar con más detalle en Deep Paint 3D, pasando al modo de Proyección.

▷ **7.** Podemos elevar el tono aumentando su escala mediante el efecto "Pitch".

▷ **8.** Aplicando un "Fade Out" en la porción final del sonido previamente seleccionada.

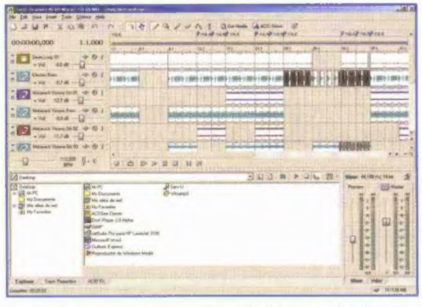
▷ **9.** Mediante la función "Crossfade Loop" es posible crear un loop perfecto en Sound Forge 6.0.

▷ **10.** Seleccionando con el ratón los que queremos añadir a la lista. Seguidamente seleccionamos la opción "Special \ Region List \ Insert". Y finalmente, asignamos un nombre a la selección.

Contenido CD-ROM 9

▶ AUDIO

■ **Acid Music 3.0**



Completo programa de edición musical con el que podrás realizar auténticas virguerías.

■ **MixMeister Pro 4.0**

Usa tus archivos MP3 para crear remixes profesionales como un DJ.

■ **Screenblast Creation Suite 1.0**

Completa suite de creación de contenidos multimedia.

■ **Atomix MP3 2.1**

Remezcla MP3 en tu PC en tiempo real y usa algoritmos de cálculo de BPM.

■ **Quartet X2 Music Studio 2.0**

Programa para crear audio en 12 canales (8 de instrumentos y 4 de percusión) fácil y rápidamente.

■ **Cool Edit Pro 2.0**

Nueva oportunidad para conseguir uno de los editores de audio más utilizado por la gran profesionalidad de los resultados obtenidos.

▶ DISEÑO 2D

■ **Adobe Illustrator 10.0**



Demo del excelente software de Adobe con el que podrás crear ilustraciones muy profesionales.

■ **Liquidpaint 1.0**

Crea psicodélicos fondos con esta sencilla herramienta.

■ **ColorExact 1.0**

Captura los colores exactos de cualquier aplicación.

■ **CompuPic Express 6.2**

Con este programa podrás crear vistas previas de todas tus imágenes, para poder localizarlas rápidamente.

▶ DISEÑO 3D

■ **Effect3D Studio**

Crea estupendos gráficos en 3D gracias a esta herramienta.

■ **Crystal 3D**



Completo programa para crear imágenes e ilustraciones en tres dimensiones.

■ **Anything 3D Pano Viewer Pro 1.0**

Añádele vistas panorámicas a tus imágenes con esta utilidad.

■ **Polytrans 3D**

Software profesional con el que podrás crear estupendos modelos en tres dimensiones.

▶ PROGRAMACIÓN

■ **Nova Engine 0.3.0**

Completo pack de librerías y clases en C y C++ especiales para la creación de juegos.

■ **JCad 1.0**

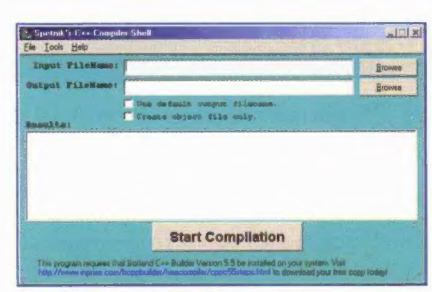
Esta utilidad te ayudará a programar modelos en 3D para tus juegos.

■ **FreeCraft Complete 1.17**

Creación de juegos de estrategia en tiempo real.

■ **Spetnik C++ Compiler Shell 3.0**

Compilador de Borland con el que podrás compilar fácilmente C++.



■ **Half-Life SDK**

Crea tus propios módulos para Half-Life, juego que adjuntamos en este CD-ROM.

▶ JUEGOS

■ **Tomb Raider**



Acompaña a Lara en su búsqueda y vive aventuras muy emocionantes.

■ **Quake**

Demo del popular juego que tan buenos momentos nos ha hecho pasar.

■ **Half-Life**

Intenta sobrevivir en esta experiencia alienígena desde la perspectiva de un soldado.

■ **Zone of Fighters**

Como cada semana, nuestra nueva versión del juego Zone of Fighters.

▶ VÍDEO

■ **NewPlay 4**

Podrás reproducir todo tipo de formatos de vídeo usando este programa.

■ **Dogma 1.2.2.**



Esta potente herramienta es capaz de reproducir múltiples formatos de vídeo y combinarlos con MP3, televisión, etc...

■ **FX Movie Splitter 4.5.10**

Combina distintos vídeos en uno solo.

▶ EXTRAS

En este apartado encontrarás todos los ejemplos de los que hablamos en el coleccionable, para que no pierdas detalle.